Large Scale Deep Learning with TensorFlow

Jeff Dean
Google Brain Team
g.co/brain

In collaboration with many other people at Google
Google Brain project started in 2011, with a focus on pushing state-of-the-art in neural networks. Initial emphasis:

- use large datasets, and
- large amounts of computation

to push boundaries of what is possible in perception and language understanding
Overview

- Cover our experience from past ~5 years
  - **Research**: speech, images, video, robotics, language understanding, NLP, translation, optimization algorithms, unsupervised learning, ...
  - **Production**: deployed systems for advertising, search, GMail, Photos, Maps, YouTube, speech recognition, image analysis, user prediction, ...

- Focus on neural nets, but many techniques more broadly applicable
What is the Google Brain Team?

- Research team focused on long term artificial intelligence research
  - Mix of computer systems and machine learning research expertise
  - Pure ML research, and research in context of emerging ML application areas:
    - robotics, language understanding, healthcare, ...

[g.co/brain](https://g.co/brain)
We Disseminate Our Work in Many Ways

• By publishing our work
  ○ See papers at research.google.com/pubs/BrainTeam.html

• By releasing TensorFlow, our core machine learning research system, as an open-source project

• By releasing implementations of our research models in TensorFlow

• By collaborating with product teams at Google to get our research into real products
What Do We Really Want?

- Build artificial intelligence algorithms and systems that learn from experience
- Use those to solve difficult problems that benefit humanity
What do I mean by understanding?
What do I mean by understanding?
What do I mean by understanding?
What do I mean by understanding?

Query

[ car parts for sale ]
What do I mean by understanding?

Query

[ car parts for sale ]

Document 1

… car parking available for a small fee.
… parts of our floor model inventory for sale.

Document 2

Selling all kinds of automobile and pickup truck parts, engines, and transmissions.
Example Needs of the Future

- Which of these eye images shows symptoms of diabetic retinopathy?
- Find me all rooftops in North America
- Describe this video in Spanish
- Find me all documents relevant to reinforcement learning for robotics and summarize them in German
- Find a free time for everyone in the Smart Calendar project to meet and set up a videoconference
- Robot, please fetch me a cup of tea from the snack kitchen
Growing Use of Deep Learning at Google

Across many products/areas:
- Android
- Apps
- drug discovery
- Gmail
- Image understanding
- Maps
- Natural language understanding
- Photos
- Robotics research
- Speech
- Translation
- YouTube
- ... many others ...

# of directories containing model description files

Unique project directories

Time

- 2012-Q1
- 2013-Q2
- 2014-Q3
- 2015-Q4
- 2016-Q1
- 2017-Q2
Overview

- Discuss TensorFlow, an open source machine learning system
  - Our primary research and production system
  - Show real examples
  - Explain what’s happening underneath the covers
Two Generations of Distributed ML Systems

1st generation - DistBelief (Dean et al., NIPS 2012)

- Scalable, good for production, but not very flexible for research

2nd generation - TensorFlow (see tensorflow.org and whitepaper 2015, tensorflow.org/whitepaper2015.pdf)

- Scalable, good for production, but also flexible for variety of research uses
- Portable across range of platforms
- Open source w/ Apache 2.0 license
Important Property of Neural Networks

Results get better with

more data +
bigger models +
more computation
Fig. 2. Performance of 5-grams against \textit{nstate} 2048 RNNs with increasing training data size. We test on Randomly Selected (RS) splits and Entropy Filtered (EF) splits of the 8bn corpus.

Fig. 3. Scaling \textit{nstate} trained on 1bn words of the entropy filtered 8bn corpus. Dashed line is the 5-gram baseline.
Large Datasets + Powerful Models

- Combination works incredibly well
- Poses interesting systems problems, though:
  - Need lots of computation
  - Want to train and do experiments quickly
  - Large-scale parallelism using distributed systems really only way to do this at very large scale
  - Also want to easily express machine learning ideas
Basics of Deep Learning

- Unsupervised cat
- Speech
- Vision

General trend is towards more complex models:
- Embeddings of various kinds
- Generative models
- Layered LSTMs
- Attention
Learning from Unlabeled Images

- Train on 10 million images (YouTube)
- 1000 machines (16,000 cores) for 1 week.
- 1.15 billion parameters
Learning from Unlabeled Images

Top 48 stimuli from the test set

Optimal stimulus by numerical optimization
Learning from Unlabeled Images

Top 48 stimuli from the test set

Optimal stimulus by numerical optimization
Adding Supervision

Top stimuli for selected neurons.
Speech: Feedforward Acoustic Models

Model speech frame-by-frame, independently

Simple fully-connected networks

Deep Neural Networks for Acoustic Modeling in Speech Recognition
CLDNNs

Model frequency invariance using 1D convolutions

Model time dynamics using an LSTM

Use fully connected layers on top to add depth

Convolutional, Long Short-Term Memory, Fully Connected Deep Neural Networks
Sainath et al. ICASSP’15
Trend: LSTMs end-to-end!

Train recurrent models that also incorporate **Lexical** and **Language Modeling:**

- Fast and Accurate Recurrent Neural Network Acoustic Models for Speech Recognition, H. Sak *et al.* 2015
- Deep Speech: Scaling up end-to-end speech recognition, A. Hannun *et al.* 2014
- Listen, Attend and Spell, W. Chan *et al.* 2015
CNNs for Vision: AlexNet

ImageNet Classification with Deep Convolutional Neural Networks
Krizhevsky, Sutskever and Hinton, NIPS 2012
The Inception Architecture (GoogLeNet, 2015)

Basic module, which is then replicated many times
The Inception Architecture (GoogLeNet, 2015)

Going Deeper with Convolutions

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich

ArXiv 2014, CVPR 2015
Rethinking the Inception Architecture for Computer Vision

Christian Szegedy
Google Inc.
szegedy@gmail.com

Vincent Vanhoucke
vanhoucke@google.com

Sergey Ioffe
sioffe@google.com

Jonathon Shlens
shlens@gmail.com

Zbigniew Wojna
University College London
zbigniewwojna@gmail.com

http://arxiv.org/abs/1512.00567
Rapid Progress in Image Recognition

<table>
<thead>
<tr>
<th>Team</th>
<th>Year</th>
<th>Place</th>
<th>Error (top-5)</th>
<th>Params</th>
</tr>
</thead>
<tbody>
<tr>
<td>XRCE (pre-neural-net explosion)</td>
<td>2011</td>
<td>1st</td>
<td>25.8%</td>
<td></td>
</tr>
<tr>
<td>Supervision (AlexNet)</td>
<td>2012</td>
<td>1st</td>
<td>16.4%</td>
<td>60M</td>
</tr>
<tr>
<td>Clarifai</td>
<td>2013</td>
<td>1st</td>
<td>11.7%</td>
<td>65M</td>
</tr>
<tr>
<td>MSRA</td>
<td>2014</td>
<td>3rd</td>
<td>7.35%</td>
<td></td>
</tr>
<tr>
<td>VGG</td>
<td>2014</td>
<td>2nd</td>
<td>7.32%</td>
<td>180M</td>
</tr>
<tr>
<td>GoogLeNet (Inception)</td>
<td>2014</td>
<td>1st</td>
<td>6.66%</td>
<td>5M</td>
</tr>
<tr>
<td>Andrej Karpathy (human)</td>
<td>2014</td>
<td>N/A</td>
<td>5.1%</td>
<td>100 trillion?</td>
</tr>
<tr>
<td>BN-Inception (Arxiv)</td>
<td>2015</td>
<td>N/A</td>
<td>4.9%</td>
<td>13M</td>
</tr>
<tr>
<td>Inception-v3 (Arxiv)</td>
<td>2015</td>
<td>N/A</td>
<td>3.46%</td>
<td>25M</td>
</tr>
</tbody>
</table>

Models with small number of parameters fit easily in a mobile app (8-bit fixed point)

ImageNet challenge classification task
What do you want in a machine learning system?

- **Ease of expression**: for lots of crazy ML ideas/algorithms
- **Scalability**: can run experiments quickly
- **Portability**: can run on wide variety of platforms
- **Reproducibility**: easy to share and reproduce research
- **Production readiness**: go from research to real products
Open, standard software for general machine learning

Great for Deep Learning in particular

First released Nov 2015

Apache 2.0 license

http://tensorflow.org/
and
https://github.com/tensorflow/tensorflow
TensorFlow:
Large-Scale Machine Learning on Heterogeneous Distributed Systems
(Preliminary White Paper, November 9, 2015)

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro,
Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow,
Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser,
Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray,
Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar,
Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernando Viégas, Oriol Vinyals,
Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng
Google Research*

Abstract

TensorFlow [1] is an interface for expressing machine learning algorithms, and an implementation for executing such algorithms. A computation expressed using TensorFlow can be executed with little or no change on a wide variety of heterogeneous systems, ranging from mobile devices such as phones

TensorFlow: A system for large-scale machine learning

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng

Google Brain

Preprint: arxiv.org/abs/1605.08695
Updated version to appear in OSDI 2016
Strong External Adoption

Adoption of Deep Learning Tools on GitHub

TensorFlow
- GitHub Stars: 27,775
- GitHub Forks: 11,220
- GitHub Launch Nov. 2015

Caffe
- GitHub Stars: 11,178
- GitHub Forks: 6,730
- GitHub Launch Sep. 2013

Torch
- GitHub Stars: 4,930
- GitHub Forks: 1,406
- GitHub Launch Jan. 2012

Theano
- GitHub Stars: 4,096
- GitHub Forks: 1,473
- GitHub Launch Jan. 2008

50,000+ binary installs in 72 hours, 500,000+ since November, 2015
Strong External Adoption

Adoption of Deep Learning Tools on GitHub

- TensorFlow: 27775 GitHub Stars, 11220 GitHub Forks, GitHub Launch Nov. 2015
- Caffe: 11178 GitHub Stars, 6730 GitHub Forks, GitHub Launch Sep. 2013
- Theano: 4096 GitHub Stars, 1473 GitHub Forks, GitHub Launch Jan. 2008

50,000+ binary installs in 72 hours, 500,000+ since November, 2015
Most forked new repo on GitHub in 2015 (despite only being available in Nov, ‘15)
Bloomberg Writes About Open Source Deep Learning Packages?

TensorFlow Mechanics 101

This is a technical tutorial, where we walk you through the details of using TensorFlow infrastructure to train models at scale. We use again MNIST as the example.

View Tutorial

Convolutional Neural Networks

An introduction to convolutional neural networks using the CIFAR-10 data set. Convolutional neural nets are particularly tailored to images, since they exploit translation invariance to yield more compact and effective representations of visual content.

View Tutorial

Vector Representations of Words

This tutorial motivates why it is useful to learn to represent words as vectors (called word embeddings). It introduces the word2vec model as an efficient method for learning embeddings. It also covers the high-level details behind noise-contrastive training methods (the biggest recent advance in training embeddings).

View Tutorial

Recurrent Neural Networks

An introduction to RNNs, wherein we train an LSTM network to predict the next word in an English sentence. (A task sometimes called language modeling.)

View Tutorial

Sequence-to-Sequence Models

A follow-on to the RNN tutorial, where we assemble a sequence-to-sequence model for machine translation. You will learn to build your own English-to-French translator, entirely machine learned, end-to-end.
Motivations

● DistBelief (our 1st system) was the first scalable deep learning system, but not as flexible as we wanted for research purposes
● Better understanding of problem space allowed us to make some dramatic simplifications
TensorFlow: Expressing High-Level ML Computations

- Core in C++
  - Very low overhead
TensorFlow: Expressing High-Level ML Computations

- Core in C++
  - Very low overhead
- Different front ends for specifying/driving the computation
  - Python and C++ today, easy to add more
TensorFlow: Expressing High-Level ML Computations

- Core in C++
  - Very low overhead
- Different front ends for specifying/driving the computation
  - Python and C++ today, easy to add more
Computation is a dataflow graph

Graph of Nodes, also called Operations or ops.
Computation is a dataflow graph

Edges are N-dimensional arrays: Tensors
Example TensorFlow fragment

- Build a graph computing a neural net inference.

```python
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data

mnist = input_data.read_data_sets('MNIST_data', one_hot=True)
x = tf.placeholder("float", shape=[None, 784])
W = tf.Variable(tf.zeros([784,10]))
b = tf.Variable(tf.zeros([10]))
y = tf.nn.softmax(tf.matmul(x, W) + b)
```
Computation is a dataflow graph

'Biases' is a variable

Some ops compute gradients

\(-=\) updates biases

biases

Add

learning rate

Mul

with state
Symbolic Differentiation

- Automatically add ops to calculate symbolic gradients of variables w.r.t. loss function.
- Apply these gradients with an optimization algorithm

```python
y_ = tf.placeholder(tf.float32, [None, 10])
cross_entropy = -tf.reduce_sum(y_ * tf.log(y))
opt = tf.train.GradientDescentOptimizer(0.01)
train_op = opt.minimize(cross_entropy)
```
Define graph and then execute it repeatedly

- Launch the graph and run the training ops in a loop

```python
init = tf.initialize_all_variables()
sess = tf.Session()
sess.run(init)
for i in range(1000):
    batch_xs, batch_ys = mnist.train.next_batch(100)
sess.run(train_step, feed_dict={x: batch_xs, y_: batch_ys})
```
Computation is a dataflow graph

```
biases

...  

learning rate

Add  

...  

Mul

GPU 0

Assign Sub

CPU
```

distributed
Assign Devices to Ops

- TensorFlow inserts `Send/Recv` Ops to transport tensors across devices
- `Recv` ops pull data from `Send` ops

TensorFlow inserts `Send/Recv` Ops to transport tensors across devices. `Recv` ops pull data from `Send` ops.
Assign Devices to Ops

- TensorFlow inserts Send/Recv Ops to transport tensors across devices
- Recv ops pull data from Send ops
Send and Receive Implementations

- Different implementations depending on source/dest devices
- e.g. GPUs on same machine: local GPU $\rightarrow$ GPU copy
- e.g. CPUs on different machines: cross-machine RPC
- e.g. GPUs on different machines: RDMA
November 2015

Release 0.5.0

Initial release of TensorFlow.
Release 0.6.0

Major Features and Improvements

- **Python 3.3+ support** via changes to python codebase and ability to specify python version via ./configure.
- **Some improvements to GPU performance** and memory usage: convnet benchmarks roughly equivalent with native cudnn v2 performance. Improvements mostly due to moving to 32-bit indices, faster shuffling kernels. More improvements to come in later releases.

Bug Fixes

- Lots of fixes to documentation and tutorials, many contributed by the public.
- 271 closed issues on github issues.
February 2016

Release 0.7.0

Major Features and Improvements

- Allow using any installed Cuda >= 7.0 and cuDNN >= R2, and add support for cuDNN R4
- Added a `contrib/` directory for unsupported or experimental features, including higher-level layers module
- Added an easy way to add and dynamically load user-defined ops
- Built out a good suite of tests, things should break less!
- Added MetaGraphDef which makes it easier to save graphs with metadata
- Added assignments for "Deep Learning with TensorFlow" Udacity course

Bug Fixes and Other Changes

- Added a versioning framework for `GraphDef`s to ensure compatibility
- Enforced Python 3 compatibility
- Internal changes now show up as sensibly separated commits
- Open-sourced the doc generator
Release 0.8.0

Major Features and Improvements

• Added a distributed runtime using GRPC
• Move skflow to contrib/learn
• Better linear optimizer in contrib/linear_optimizer
• Random forest implementation in contrib/tensor_forest
• CTC loss and decoders in contrib/ctc
• Basic support for half data type
• Better support for loading user ops (see examples in contrib/)
• Allow use of (non-blocking) Eigen threadpool with TENSORFLOW_USE_EIGEN_THREADPOOL define
• Add an extension mechanism for adding network file system support
• TensorBoard displays metadata stats (running time, memory usage and device used) and tensor shapes

Big Fixes and Other Changes

• Utility for inspecting checkpoints
• Basic tracing and timeline support
• Allow building against cuDNN 5 (not incl. RNN/LSTM support)
June 2016

Release 0.9.0

Major Features and Improvements

- Python 3.5 support and binaries
- Added iOS support
- Added support for processing on GPUs on MacOS
- Added makefile for better cross-platform build support (C API only)
- fp16 support and improved complex128 support for many ops
- Higher level functionality in contrib.layers,losses,metrics,learn
- More features to Tensorboard
- Improved support for string embedding and sparse features
- The RNN api is finally "official" (see, e.g., tf.nn.dynamic_rnn, tf.nn.rnn, and the classes in tf.nn.rnn_cell).
- TensorBoard now has an Audio Dashboard, with associated audio summaries.

Big Fixes and Other Changes

- Turned on CuDNN Autotune.
- Added support for using third-party Python optimization algorithms (contrib.opt).
- Google Cloud Storage filesystem support.
- HDF5 support.
Activity

6,354 commits
12 branches
10 releases
335 contributors

Contributions to master, excluding merge commits
DeepMind moves to TensorFlow

Friday, April 29, 2016

Posted by Koray Kavukcuoglu, Research Scientist, Google DeepMind

At DeepMind, we conduct state-of-the-art research on a wide range of algorithms, from deep learning and reinforcement learning to systems neuroscience, towards the goal of building Artificial General Intelligence. A key factor in facilitating rapid progress is the software environment used for research. For nearly four years, the open source Torch7 machine learning library has served as our primary research platform, combining excellent flexibility with very fast runtime execution, enabling rapid prototyping. Our team has been proud to contribute to the open source project in capacities ranging from occasional bug fixes to being core maintainers of several crucial components.

With Google’s recent open source release of TensorFlow, we initiated a project to test its suitability for our research environment. Over the last six months, we have re-implemented more than a dozen different projects in TensorFlow to develop a deeper understanding of its potential use cases and the tradeoffs for research. Today we are excited to announce that DeepMind will start using TensorFlow for all our future research. We believe that TensorFlow will enable us to execute our ambitious research goals at much larger scale and an even faster pace, providing us with a unique opportunity to further accelerate our research programme.

As one of the core contributors of Torch7, I have had the pleasure of working closely with an excellent community of developers and researchers, and it has been amazing to see all the great work that has been achieved with the library. TensorFlow is an incredibly exciting tool for deep learning and we are excited to bring this technology to our research environment.
Dear TensorFlow community,

Today we are releasing our best image classifier trained on ImageNet data. As described in our recent Arxiv preprint at http://arxiv.org/abs/1512.00567, an ensemble of four of these models achieves 3.5% top-5 error on the validation set of the ImageNet whole image ILSVRC2012 classification task (compared with our ensemble from last year that won the 2014 ImageNet classification challenge with a 6.66% top-5 error rate).

In this release, we are supplying code and data files containing the trained model parameters for running the image classifier on:

- Both desktop and mobile environments
- Employing either a C++ or Python API.

In addition, we are providing a tutorial that describes how to use the image recognition system for a variety of use-cases.

http://www.tensorflow.org/tutorials/image_recognition/index.html
bazel build tensorflow/examples/label_image/...

That should create a binary executable that you can then run like this:

bazel-bin/tensorflow/examples/label_image/label_image

This uses the default example image that ships with the framework and should output something similar to this:

```
I tensorflow/examples/label_image/main.cc:200] military uniform (656): 0.647296
I tensorflow/examples/label_image/main.cc:200] suit (794): 0.8477196
I tensorflow/examples/label_image/main.cc:200] academic gown (606): 0.0232411
I tensorflow/examples/label_image/main.cc:200] bow tie (817): 0.8157556
I tensorflow/examples/label_image/main.cc:200] boho tie (960): 0.0148624
```

In this case, we're using the default image of Admiral Grace Hopper, and you can see the network correctly identifies she's wearing a military uniform, with a high score of 0.6.
Experiment Turnaround Time and Research Productivity

- **Minutes, Hours:**
  - Interactive research! Instant gratification!

- **1-4 days**
  - Tolerable
  - Interactivity replaced by running many experiments in parallel

- **1-4 weeks**
  - High value experiments only
  - Progress stalls

- **>1 month**
  - Don’t even try
Data Parallelism

● Use multiple model replicas to process different examples at the same time
  ○ All collaborate to update model state (parameters) in shared parameter server(s)

● Speedups depend highly on kind of model
  ○ Dense models: 10-40X speedup from 50 replicas
  ○ Sparse models:
    ■ support many more replicas
    ■ often can use as many as 1000 replicas
Data Parallelism

Parameter Servers

Model Replicas

Data
Data Parallelism

Parameter Servers

Model Replicas

Data
Data Parallelism

Parameter Servers

$p$

$\Delta p$

Model Replicas

Data
Data Parallelism

\[ p' = p + \Delta p \]

Parameter Servers

Model Replicas

Data

\[ \Delta p \]

\[ p \]
Data Parallelism

$p' = p + \Delta p$

Parameter Servers

Model Replicas

Data
Data Parallelism

Parameter Servers

Δp' → p'

Model Replicas

Data
Data Parallelism

\[ p'' = p' + \Delta p \]

Parameter Servers

\[ \Delta p' \]

Model Replicas

\[ p' \]

Data
Data Parallelism

Parameter Servers

\[ p'' = p' + \Delta p \]

Model Replicas

\[ \Delta p' \]

\[ p' \]

Data
DistBelief: Separate Parameter Servers

Parameter update rules not the same programming model as the rest of the system

Separate code for parameter servers vs. rest of system

Lacked uniformity & was more complicated
Cross process communication is the same!

- Communication across machines over the network abstracted identically to cross device communication.

No specialized parameter server subsystem!
Data Parallelism Choices

Can do this **synchronously**:

- **N replicas** equivalent to an **N times larger batch size**
- Pro: No gradient staleness
- Con: Less fault tolerant (requires some recovery if any single machine fails)

Can do this **asynchronously**:

- Pro: Relatively fault tolerant (failure in model replica doesn’t block other replicas)
- Con: Gradient staleness means each gradient less effective

(Or **hybrid**: M asynchronous groups of N synchronous replicas)
Asynchronous Training

- Unlike DistBelief, no separate parameter server system:
  - Parameters are now just stateful nodes in the graph
Synchronous Variant

Parameter Device(s)

Client

Add → Update → P

ΔP

Device A

- model
- input

Device B

- model
- input

Device C

- model
- input

Synchronous Data Parallelism
Synchronous vs. Asynchronous

Graph structure and low-level graph primitives (queues) allow us to play with synchronous vs. asynchronous update algorithms.
Data Parallelism Considerations

Want model computation time to be large relative to time to send/receive parameters over network

Models with fewer parameters, that reuse each parameter multiple times in the computation

- Mini-batches of size $B$ reuse parameters $B$ times

 Certain model structures **reuse each parameter** many times within each example:

- **Convolutional models** tend to reuse hundreds or thousands of times per example (for different spatial positions)
- **Recurrent models** (LSTMs, RNNs) tend to reuse tens to hundreds of times (for unrolling through $T$ time steps during training)
Success of Data Parallelism

- Data parallelism is **really important** for many of Google’s problems (very large datasets, large models):
  - RankBrain uses 500 replicas
  - ImageNet Inception training uses 50 GPUs, ~40X speedup
  - SmartReply uses 16 replicas, each with multiple GPUs
  - State-of-the-art on LM “One Billion Word” Benchmark model uses both data and model parallelism on 32 GPUs
Image Model Training Time

Precision @ 1

- 50 GPUs
- 10 GPUs
- 1 GPU

Hours
Image Model Training Time

Precision @ 1

2.6 hours vs. 79.3 hours (30.5X)

50 GPUs
10 GPUs
1 GPU

Hours
Synchronous converges faster (time to accuracy)

Test accuracy

Synchronous updates (with backup workers) trains to higher accuracy faster
Better scaling to more workers (less loss of accuracy)

Synchronous converges faster (time to accuracy)

Synchronous updates (with backup workers) trains to higher accuracy faster
Better scaling to more workers (less loss of accuracy)

Synchronous converges faster (time to accuracy)

Synchronous updates (with backup workers) trains to higher accuracy faster
Better scaling to more workers (less loss of accuracy)

Synchronous converges faster (time to accuracy)

Synchronous updates (with backup workers) trains to higher accuracy faster
Better scaling to more workers (less loss of accuracy)

General Computations

Although we originally built TensorFlow for our uses around deep neural networks, it’s actually quite flexible.

Wide variety of machine learning and other kinds of numeric computations easily expressible in the computation graph model.
Runs on Variety of Platforms

phones

single machines (CPU and/or GPUs) ...

distributed systems of 100s of machines and/or GPU cards

custom ML hardware
Trend: Much More Heterogeneous hardware

General purpose CPU performance scaling has slowed significantly

Specialization of hardware for certain workloads will be more important
Tensor Processing Unit

Custom machine learning ASIC

In production use for >16 months: used on every search query, used for AlphaGo match, many other uses, ...

Extensible

- Core system defines a number of standard *operations* and *kernels* (device-specific implementations of operations)
- Easy to define new operators and/or kernels
A tour through the TensorFlow codebase

1. Expressing graphs

Slide credit: Kevin Robinson (krob@mit.edu)

core: graph, ops, protobuf

core: graph, ops, protobuf

python: variables, optimizer

http://public.kevinrobinsonblog.com/tensorflow-codebase/
Expressing: Graphs and Ops

Graph

Slide credit: Kevin Robinson (krob@mit.edu)
Expressing: Graphs and Ops

Graph

Ops

Slide credit: Kevin Robinson (krob@mit.edu)
Expressing: Graphs and Ops

```python
import tensorflow as tf

b = tf.Variable(tf.zeros([100]))
W = tf.Variable(tf.random_uniform([784, 100], -1, 1))
x = tf.placeholder(tf.float32, name="x")
relu = tf.nn.relu(tf.matmul(W, x) + b)
cost = # ...

s = tf.Session()
for step in xrange(0, 10):
    input = # ...read in 100-D input array ...
    result = s.run(cost, feed_dict={x: input})
print step, result
```
Expressing: Ops

```python
import tensorflow as tf

b = tf.Variable(tf.zeros([100]))
W = tf.Variable(tf.random_uniform([784, 100], -1, 1))
x = tf.placeholder(tf.float32, name="x")
relu = tf.nn.relu(tf.matmul(W, x) + b)

cost = # ...

s = tf.Session()
for step in xrange(0, 10):
    input = # ...read in 100-D input array ...
    result = s.run(cost, feed_dict={x: input})
    print step, result
```
Expressing: Ops

```
import tensorflow as tf

b = tf.Variable(tf.zeros([100]))
W = tf.Variable(tf.random_uniform([784,100],[-1,1]))
x = tf.placeholder(tf.float32, name="x")
relu = tf.nn.relu(tf.matmul(W, x))
cost = # ...

with tf.Session() as s:
    input = # ...read in 100-D input array ...
    result = s.run(cost, feed_dict={x: input})
    print step, result
```
Expressing: Ops

tf.matmul(W, x)

in `math_ops.py#L1137`

```
return gen_math_ops._mat_mul(a, b,
    transpose_a=transpose_a,
    transpose_b=transpose_b,
    name=name)
```

calls C++ wrappers generated by `cc/BUILD#L27`

**OpDef** interface defined in `math_ops.cc#L607`

```
REGISTER_OP("MatMul")
  .Input("a: T")
  .Input("b: T")
  .Output("product: T")
  .Attr("transpose_a: bool = false")
  .Attr("transpose_b: bool = false")
  .Attr("T: {float, double, int32, complex64}")
```
Expressing: Graph

```python
import tensorflow as tf

b = tf.Variable(tf.zeros([100]))
W = tf.Variable(tf.random_uniform([784, 100], -1, 1))
x = tf.placeholder(tf.float32, name="x")
relu = tf.nn.relu(tf.matmul(tf.matmul(W, x))
cost = # ...

s = tf.Session()
for step in xrange(0, 10):
    input = # ...read in 100-D input array ...
    result = s.run(cost, feed_dict={x: input})
    print step, result
```
import tensorflow as tf

b = tf.Variable(tf.zeros([100]))
W = tf.Variable(tf.random_uniform([784, 100], -1, 1))
x = tf.placeholder(tf.float32, name="x")
relu = tf.nn.relu(tf.matmul(W, x) + b)
cost = # ...

s = tf.Session()
for step in xrange(0, 10):
    input = # ... read in 100-D input array ...
    result = s.run(cost, feed_dict={x: input})
    print step, result
Expressing: Graph

Graph is built implicitly

```python
tf.matmul(W, x)
print(tf.get_default_graph().as_graph_def())
```

Slide credit: Kevin Robinson (krob@mit.edu)
Expressing: Graph

Graph is built implicitly
session.py#L896

Variables add implicit ops
variables.py#L146

```python
tf.matmul(W, x)
print(tf.get_default_graph().as_graph_def())
```

```python
W = tf.Variable(tf.random_uniform([784, 100], -1, 1))
print(tf.get_default_graph().as_graph_def())
```
Expressing: Graph

Graph is built implicitly
`session.py#L896`

Variables add implicit ops
`variables.py#L146`

```
tf.matmul(W, x)
print(tf.get_default_graph().as_graph_def())
```

```
W = tf.Variable(tf.random_uniform([784, 100], -1, 1))
print(tf.get_default_graph().as_graph_def())
```

In TensorBoard:
Expressing: Optimizers

Optimizer fns extend the graph

```
import tensorflow as tf

# Create a TensorFlow graph

# Define the optimization function
optimizer = tf.train.GradientDescentOptimizer(0.01)

# Execute the optimization function
train_step = optimizer.minimize(cross_entropy)
```

Slide credit: Kevin Robinson (krob@mit.edu)
Expressing: Optimizers

Optimizer fns extend the graph
optimizer.py::minimize#L155

Trainable variables collected
variables.py#L258

Slide credit: Kevin Robinson (krob@mit.edu)
Expressing: Optimizers

- Optimizer fns extend the graph
  `optimizer.py:minimize#L155`

- Trainable variables collected
  `variables.py#L258`

- Graph is extended with gradients
  `gradients.py#L307`

```python
optimizer = tf.train.GradientDescentOptimizer(0.01)
train_step = optimizer.minimize(cross_entropy)
```
Expressing: Graph

Serialized as GraphDef

```
print(tf.get_default_graph().as_graph_def())
```

Slide credit: Kevin Robinson (krob@mit.edu)
Expressing: Graph

Serialized as GraphDef

```
print(tf.get_default_graph().as_graph_def())
```

Slide credit: Kevin Robinson (krob@mit.edu)
Expressing: Graph

Serialized as GraphDef graph.proto

```python
print(tf.get_default_graph().as_graph_def())
```

Slide credit: Kevin Robinson (krob@mit.edu)
Expressing: Graph

Serialized as GraphDef

```python
print(tf.get_default_graph().as_graph_def())
```

```
node {
  name: "MatMul"
  op: "MatMul"
  input: "W/read"
  input: "x"
  attr {
    key: "T"
    value {
      type: DT_FLOAT
    }
  }
}
node {
  name: "add"
  op: "Add"
  input: "MatMul"
  input: "b/read"
  attr {
    key: "transpose_a"
    value {
      b: false
    }
  }
}
node {
  name: "Relu"
  op: "Relu"
  input: "add"
  attr {
    key: "T"
    value {
      type: DT_FLOAT
      b: false
    }
  }
}
```
Expressing: Graph

Serialized as GraphDef

graph.proto

Slide credit: Kevin Robinson (krob@mit.edu)
Distributing

- Sessions in distributed runtime
- Pruning
- Placing and Partitioning
Distributing: Creating a session

Slide credit: Kevin Robinson (krob@mit.edu)
Distributing: Creating a session

tf.Session

gRPC: MasterService

worker process 1

worker process 2

worker process 3

Slide credit: Kevin Robinson (krob@mit.edu)
Distributing: Creating a session

```
import tensorflow as tf

b = tf.Variable(tf.zeros([100]))
W = tf.Variable(tf.random_uniform([784, 100], -1, 1))
x = tf.placeholder(tf.float32, name="x")
relu = tf.nn.relu(tf.matmul(W, x) + b)
cost = # ...

s = tf.Session()
```
Distributing: Creating a session

```python
import tensorflow as tf

b = tf.Variable(tf.zeros([100]))
W = tf.Variable(tf.random_uniform([784, 100], -1, 1))
x = tf.placeholder(tf.float32, name="x")
relu = tf.nn.relu(tf.matmul(W, x) + b)
cost = # ...
s = tf.Session()
```

Slide credit: Kevin Robinson (krob@mit.edu)
Distributing: Creating a session

```python
import tensorflow as tf
defining_code_blocks
```

Slide credit: Kevin Robinson (krob@mit.edu)
Distributing: Creating a session

```python
import tensorflow as tf

b = tf.Variable(tf.zeros([100]))
W = tf.Variable(tf.random_uniform([784, 100], -1, 1))
x = tf.placeholder(tf.float32, name="x")
relu = tf.nn.relu(tf.matmul(W, x) + b)
cost = # ...

s = tf.Session()
```

Slide credit: Kevin Robinson (krob@mit.edu)
Distributing: Running a session

```
result = s.run(cost, feed_dict={x: input})
```

- **tf.Session**
- **gRPC: Session**
  - CreateSession(GraphDef)

Slide credit: Kevin Robinson (krob@mit.edu)
Distributing: Running a session

```
result = s.run(cost, feed_dict={x: input})
```

- `tf.Session` 
- `gRPC: MasterService` 
- `CreateSession(GraphDef)` 
- `RunStep(feed, fetches)` 

Slide credit: Kevin Robinson (krob@mit.edu)
Distributing: Running a session

```
Distributing: Running a session

tf.Session

createSession(GraphDef)
runStep(feed, fetches)

gRPC: MasterService

createSession(GraphDef)
runStep(feed, fetches)

WorkerService
/job:worker/task:0

WorkerService
/job:worker/task:1

WorkerService
/job:worker/task:2

execute subgraph

Slide credit: Kevin Robinson (krob@mit.edu)
```
Distributing: Running a session

```
tf.Session
```

```
gRPC: MasterService
CreateSession(GraphDef)
RunStep(feed, fetches)
```

```
gRPC: Session
```

```
WorkerService
/job:worker/task:0
```

```
WorkerService
/job:worker/task:1
```

```
WorkerService
/job:worker/task:2
```

Slide credit: Kevin Robinson (krob@mit.edu)
Distributing: Running a session

- Distribute session creation and execution across multiple workers

**tf.Session**

**gRPC:**
- MasterService: CreateSession(GraphDef), RunStep(feed, fetches)

**WorkerService**
- /job:worker/task:0
  - RunGraph(graph, feed, fetches)
- /job:worker/task:1
  - RunGraph(graph, feed, fetches)
- /job:worker/task:2
  - RunGraph(graph, feed, fetches)

Hardware:
- CPU
- GPU

Slide credit: Kevin Robinson (krob@mit.edu)
Distributing: Running a session

```
Distributing: Running a session

- tf.Session
- gRPC: MasterService
  - CreateSession(GraphDef)
  - RunStep(feed, fetches)
- WorkerService
  - /job:worker/task:0
    - RunGraph(graph, feed, fetches)
    -RecvTensor(rendezvous_key)
  - /job:worker/task:1
    - RunGraph(graph, feed, fetches)
    -RecvTensor(rendezvous_key)
  - /job:worker/task:2
    - RunGraph(graph, feed, fetches)
    -RecvTensor(rendezvous_key)

Slide credit: Kevin Robinson (krob@mit.edu)
```
Distributing: Running a session

```python
result = s.run(cost, feed_dict={x: input})
```

- **tf.Session**
- **gRPC: MasterService**
  - CreateSession(GraphDef)
  - RunStep(feed, fetches)
- **WorkerService**
  - /job:worker/task:0
    - RunGraph(graph, feed, fetches)
    -RecvTensor(rendezvous_key)
  - /job:worker/task:1
    - RunGraph(graph, feed, fetches)
    -RecvTensor(rendezvous_key)
  - /job:worker/task:2
    - RunGraph(graph, feed, fetches)
    -RecvTensor(rendezvous_key)

Slide credit: Kevin Robinson (krob@mit.edu)
Distributing: Pruning

gRPC call to `Session::Run` in `master_session.cc#L835`

Slide credit: Kevin Robinson (krob@mit.edu)
Distributing: Pruning

gRPC call to `Session::Run` in `master_session.cc#L835`

Rewrite with feed and fetch

`RewriteGraphForExecution` in `graph/subgraph.cc#L225`

Slide credit: Kevin Robinson (krob@mit.edu)
Distributing: Pruning

gRPC call to `Session::Run` in `master_session.cc#L835`

**Rewrite** with feed and fetch

`RewriteGraphForExecution` in `graph/subgraph.cc#L225`

**Prune** subgraph

`PruneForReverseReachability` in `graph/algorithm.cc#L122`

Tests in `subgraph_test.cc#142`

Slide credit: Kevin Robinson (krob@mit.edu)
Distributing: Placing

Constraints from model

DeviceSpec in device.py#L24

```python
with tf.device("/job:ps/task:0"):
    weights_1 = tf.Variable(...)
    biases_1 = tf.Variable(...)

with tf.device("/job:ps/task:1"):
    weights_2 = tf.Variable(...)
    biases_2 = tf.Variable(...)

with tf.device("/job:worker/task:7"):
    input, labels = ...
    layer_1 = tf.nn.relu(tf.matmul(input, weights_1)
    logits = tf.nn.relu(tf.matmul(layer_1, weights
    # ...
```
Distributing: Placing

Constraints from model

DeviceSpec in device.py#L24

By device or colocation

NodeDef in graph.proto

```python
with tf.device("/job:ps/task:0"):
    weights_1 = tf.Variable(...)  
    biases_1 = tf.Variable(...)  

with tf.device("/job:ps/task:1"):
    weights_2 = tf.Variable(...)  
    biases_2 = tf.Variable(...)  

with tf.device("/job:worker/task:7"):
    input, labels = ...
    layer_1 = tf.nn.relu(tf.matmul(input, weights_1))  
    logits = tf.nn.relu(tf.matmul(layer_1, weights_2))  
    # ...
```

graph { node { device: "" } }

Slide credit: Kevin Robinson (krob@mit.edu)
Distributing: Placing

Placing based on constraints

`SimplePlacer::Run`

in `simple_placer.cc#L558`
described in `simple_placer.h#L31`
Distributing: Placing

Placing based on constraints

`SimplePlacer::Run`

in `simple_placer.cc#L558`

described in `simple_placer.h#L31`
Distributing: Placing

Placing based on constraints

SimplePlacer::Run

in simple_placer.cc#L558

described in simple_placer.h#L31

WorkerService
/job:worker/task:0

Slide credit: Kevin Robinson (krob@mit.edu)
Distributing: Partitioning

Partition into subgraphs
in `graph_partition.cc#L883`
Distributing: Partitioning

**Partition** into subgraphs
in `graph_partition.cc#L883`

Rewrite with **Send** and **Recv**
in `sendrecv_ops.cc#L56` and `#L97`

WorkerService
/job:worker/task:0

```
Device A
```

```
Device B
```

WorkerService
/job:worker/task:0
Distributing: Partitioning

Partition into subgraphs
in `graph_partition.cc#L883`

Rewrite with **Send** and **Recv**
in `sendrecv_ops.cc#L56` and `#L97`

**Rendezvous** handles coordination
in `base_rendezvous_mgr.cc#L236`
Distributing: Partitioning

Partition into subgraphs
in `graph_partition.cc#L883`

Rewrite with **Send** and **Recv**
in `sendrecv_ops.cc#L56` and `#L97`

**Rendezvous** handles coordination
in `base_rendezvous_mgr.cc#L236`

---

Slide credit: Kevin Robinson (krob@mit.edu)
A tour through the TensorFlow codebase

2. Distributing the graph
Executing: Executor

**Parallelism** on each worker

```
WorkerService
/job:worker/task:0
    RunGraph(graph, feed, fetches)
   RecvTensor(rendezvous_key)
```

<table>
<thead>
<tr>
<th>CPU</th>
<th>GPU</th>
<th>GPU</th>
<th>GPU</th>
</tr>
</thead>
</table>

Slide credit: Kevin Robinson (krob@mit.edu)
Executing: Executor

**Parallelism** on each worker

```java
WorkerService
/job:worker/task:0

RunGraph(graph,feed,fetches)
RecvTensor(rendezvous_key)
```

<table>
<thead>
<tr>
<th>CPU</th>
<th>GPU</th>
<th>GPU</th>
<th>GPU</th>
</tr>
</thead>
</table>
Executing: Executor

**Parallelism** on each worker

```
WorkerService
/job:worker/task:0
RunGraph(graph,feed,fetches)
RecvTensor(rendezvous_key)
```

| CPU | GPU | GPU | GPU |

GraphMgr::ExecuteAsync
in `graph_mgr.cc#L283`

ExecutorState::RunAsync
in `executor.cc#L867`
Executing: OpKernels

WorkerService
/job:worker/task:0

rendezvous

Device A
Executing: OpKernels

WorkerService
/job:worker/task:0

rendezvous

Slide credit: Kevin Robinson (krob@mit.edu)
Executing: OpKernels

REGISTER_OP("MatMul")
  .Input("a: T")
  .Input("b: T")
  .Output("product: T")
  .Attr("transpose_a: bool = false")
  .Attr("transpose_b: bool = false")
  .Attr("T: {float, double, int32, complex64}'")
  .Doc("Multiply the matrix "a" by the matrix "b".")
Executing: OpKernels

WorkerService
/job:worker/task:0

rendezvous

Conv2D OpDef in nn_ops.cc#L221

```cpp
REGISTER_OP("Conv2D")
  .Input("input: T")
  .Input("filter: T")
  .Output("output: T")
  .Attr("T: {float, double}")
  .Attr("strides: list(int)")
  .Attr("use_cudnn_on_gpu: bool = true")
  .Attr("GetPaddingAttrString()")
```
Executing: OpKernels

Conditional build for OpKernels

```cpp
#if GOOGLE_CUDDA

// Registration of the GPU implementations.
REGISTER_KERNEL_BUILDER(
    Name("Conv2D").Device(DEVICE_GPU).TypeConstraint<float>("T"),
    Conv2DOp<GPUDevice, float>);
#endif  // GOOGLE_CUDDA
```
Executing: OpKernels

Conditional build for OpKernels

```cpp
#if GOOGLE_CUD

// Registration of the GPU implementations.
REGISTER_KERNEL_BUILDER(
    Name("Conv2D").Device(DEVICE_GPU).TypeConstraint<float>("T"),
    Conv2DOp<GPUDevice, float>);
#endif  // GOOGLE_CUD
```

CPU in `conv_ops.cc#L91`
GPU in `conv_ops.cc#L263`

Slide credit: Kevin Robinson (krob@mit.edu)
**Executing: OpKernels**

OpKernels are **specialized** by device

adapted from **matmul_op.cc#L116**

```cpp
template <typename Device, typename T, bool USE_CUBLAS>
class MatMulOp : public OpKernel {
    public:
        explicit MatMulOp(OpKernelConstruction* ctx) : OpKernel(ctx) {
            OP_REQUIRES_OK(ctx, ctx->GetAttr("transpose_a", &transpose_a_));
            OP_REQUIRES_OK(ctx, ctx->GetAttr("transpose_b", &transpose_b_));
        }

        void Compute(OpKernelContext* ctx) override {
            const Tensor& a = ctx->input(0);
            const Tensor& b = ctx->input(1);

            //...
            LaunchMatMul<Device, T, USE_CUBLAS>::launch(ctx, this, a, b, dim_pair, out);
}

private:

Slide credit: Kevin Robinson (krob@mit.edu)
Executing: OpKernels

OpKernels are **specialized** by device adapted from `matmul_op.cc#L116`

```cpp
template <typename Device, typename T, bool USE_CUBLAS>
class MatMulOp : public OpKernel {

public:

    explicit MatMulOp(OpKernelConstruction* ctx) : OpKernel(ctx) {
        OP_REQUIRES_OK(ctx, ctx->GetAttr("transpose_a", &transpose_a_));
        OP_REQUIRES_OK(ctx, ctx->GetAttr("transpose_b", &transpose_b_));
    }

    void Compute(OpKernelContext* ctx) override {
        const Tensor& a = ctx->input(0);
        const Tensor& b = ctx->input(1);

        //...
        LaunchMatMul<Device, T, USE_CUBLAS>::launch(ctx, this, a, b, dim_pair, out);
    }

private:
```

Slide credit: Kevin Robinson (krob@mit.edu)
Executing: OpKernels

OpKernels call into Stream functions adapted from matmul_op.cc#L71

```c++
struct LaunchMatMul<GPUDevice, T, true /* USE_CUBLAS */> {
  static void launch(..., const Tensor& a, const Tensor& b, ..., Tensor* out) {
    const uint64 m = a.dim_size(1 - dim_pair[0].first);
    const uint64 k = a.dim_size(dim_pair[0].first);
    const uint64 n = b.dim_size(1 - dim_pair[0].second);
    // ...

    // Get a Stream for this GPUDevice
    auto* stream = ctx->op_device_context<GPUDeviceContext>() -> stream();
    // ...

    // Launch the BLAS gemm kernel on the GPU stream
    bool blas_launch_status = stream->ThenBlasGemm(blas_transpose_b, blas_transpose_a,
                                                  n, m, k, 1.0f, b_ptr,
                                                  transpose_b ? k : n, a_ptr,
                                                  transpose_a ? m : k, 0.0f, &c_ptr,
                                                  n).ok();

    // ... return

}
Executing: OpKernels

OpKernels call into **Stream** functions adaptation from [matmul_op.cc#L71](#L71)

```c
struct LaunchMatMul<GPUDevice, T, true /* USE_CUBLAS */> {
    static void launch(..., const Tensor& a, const Tensor& b, ..., Tensor* out) {
        const uint64 m = a.dim_size(1 - dim_pair[0].first);
        const uint64 k = a.dim_size(dim_pair[0].first);
        const uint64 n = b.dim_size(1 - dim_pair[0].second);
        // ...
        
        // Get a Stream for this GPUDevice
        auto* stream = ctx->op_device_context<GPUDeviceContext>()->stream();
        // ...
        
        // Launch the BLAS gemm kernel on the GPU stream
        bool blas_launch_status = stream->ThenBlasGemm(blas_transpose_b, blas_transpose_a, n, m, k, 1.0f, b_ptr, transpose_b ? k : n, a_ptr, transpose_a ? m : k, 0.0f, &c_ptr, n).ok();
        // ... return
    }
};
```
Executing: OpKernels

OpKernels call into **Stream** functions
adapted from *matmul_op.cc*L71

```c
struct LaunchMatMul<GPUDevice, T, true /* USE_CUBLAS */> {
  static void launch(..., const Tensor& a, const Tensor& b, ..., Tensor* out) {
    const uint64 m = a.dim_size(1 - dim_pair[0].first);
    const uint64 k = a.dim_size(dim_pair[0].first);
    const uint64 n = b.dim_size(1 - dim_pair[0].second);
    // ...

    // Get a Stream for this GPUDevice
    auto* stream = ctx->op_device_context<GPUDeviceContext>()-&gt;stream();
    // ...

    // Launch the BLAS gemm kernel on the GPU stream
    bool blas_launch_status = stream-&gt;ThenBlasGemm(blas_transpose_b, blas_transpose_a, 
    n, m, k, 1.0f, b_ptr, 
    transpose_b ? k : n, a_ptr, 
    transpose_a ? m : k, 0.0f, &c_ptr, 
    n).ok();

    // ... return
}
```
Executing: Stream functions

OpKernels call into **Stream** functions

in [conv_ops.cc#L292](#)

```c
bool blas_launch_status =
    stream
        ->ThenBlasGemm(no_transpose, no_transpose, n, m, k, 1.0f, b_
        ->ThenBlasGemm(no_transpose, no_transpose, n, m, k, a_ptr, k, 0.0f, &c_ptr, n)
        .ok();
```

Slide credit: Kevin Robinson (krob@mit.edu)
Executing: Stream functions

OpKernels call into **Stream** functions

in `conv_ops.cc#L292`

```cpp
bool blas_launch_status =
    stream
    ->ThenBlasGemm(no_transpose, no_transpose, n, m, k, 1.0f, b_
        n, a_ptr, k, 0.0f, &c_ptr, n)
    .ok();
```

in `conv_ops.cc#L417`

```cpp
CudnnScratchAllocator scratch_allocator(ConvolveScratchSize, ctx);
bool cudnn_launch_status =
    stream
    ->ThenConvolveWithScratch(input_desc, input_ptr, filter_desc, filter_ptr, conv_desc, output_desc,
                                &output_ptr, &scratch_allocator)
    .ok();
```
Platforms provide GPU-specific implementations

cuBLAS
BlasSupport in stream_executor/blas.h#L88
DoBlasInternal in cuda_blas.cc#L429
Platforms provide GPU-specific implementations

**cuBLAS**
- BlasSupport in `stream_executor/blas.h#L88`
- DoBlasInternal in `cuda_blas.cc#L429`

**cuDNN**
- DnnSupport in `stream_executor/dnn.h#L544`
- DoConvolve in `cuda_dnn.cc#L629`

```c
status = dynload::cudnnConvolutionForward(
    parent_, ToHandle(dnn_handle_),
    /*alpha=*/&alpha, /*srcDesc=*/input_4d.handle(),
    /*srcData=*/input_data.opaque(), /*filterDesc=*/filter.handle(),
    /*filterData=*/filter_data.opaque(), /*convDesc=*/conv.handle(),
    /*algo=*/algo, /*workSpace=*/scratch.opaque(),
    /*workSpaceSizeInBytes=*/scratch.size(), /*beta=*/&beta,
    /*destDesc=*/output_4d.handle(), /*destData=*/output_data->opaque());
```
Session Interface

- **Extend**: add nodes to computation graph
- **Run**: execute an arbitrary subgraph
  - optionally feeding in Tensor inputs and retrieving Tensor output

Typically, setup a graph with one or a few `Extend` calls and then `Run` it thousands or millions or times
Single device performance important, but big...

...gest performance improvements come from large-scale distributed systems with model and data parallelism
Experiment Turnaround Time and Research Productivity

- **Minutes, Hours:**
  - Interactive research! Instant gratification!

- **1-4 days**
  - Tolerable
  - Interactivity replaced by running many experiments in parallel

- **1-4 weeks**
  - High value experiments only
  - Progress stalls

- **>1 month**
  - Don’t even try
Transition

- How do you do this at scale?
- How does TensorFlow make distributed training easy?
Model Parallelism

- Best way to decrease training time: decrease the step time
- Many models have lots of inherent parallelism
- Problem is distributing work so communication doesn’t kill you
  - local connectivity (as found in CNNs)
  - towers with little or no connectivity between towers (e.g. AlexNet)
  - specialized parts of model active only for some examples
Exploiting Model Parallelism

On a single core: Instruction parallelism (SIMD). Pretty much free.

Across cores: thread parallelism. Almost free, unless across sockets, in which case inter-socket bandwidth matters (QPI on Intel).

Across devices: for GPUs, often limited by PCIe bandwidth.

Across machines: limited by network bandwidth / latency
Model Parallelism
Model Parallelism: Partition model across machines

Layer 0

Layer 1

Layer N

Partition 1  Partition 2  Partition 3

Partition 1  Partition 2  Partition 3

Partition 1  Partition 2  Partition 3
Model Parallelism: Partition model across machines

Minimal network traffic: The most densely connected areas are on the same partition

Layer N
...
Layer 1
Layer 0
Using TensorFlow for Parallelism

Easy to express both model parallelism as well as data parallelism

- Very minimal changes to single device model code
Devices and Graph Placement

- Given a graph and set of devices, TensorFlow implementation must decide which device executes each node.
Full and Partial Device Constraints (Hints)

Devices are named hierarchically:

/job:localhost/device:cpu:0  
/job:worker/task:17/device:gpu:3  
/job:parameters/task:4/device:cpu:0

Client can specify full or partial constraints for nodes in graph:

“Place this node on /job:localhost/device:gpu:2”

“Place this node on /device:gpu:*”
Placement Algorithm

Given hints, plus a cost model (node execution time estimates and Tensor size estimates), make placement decisions

- Current relatively simple greedy algorithm
- Active area of work

Show CIFAR10 placement TensorBoard.
Google Photos Search
Reuse same model for completely different problems

Same basic model structure trained on different data, useful in completely different contexts

Example: given image → predict interesting pixels
We have tons of vision problems

Image search, StreetView, Satellite Imagery, Translation, Robotics, Self-driving Cars,

www.google.com/sunroof

1234 Bryant St, Palo Alto, CA 94301, USA

Analysis complete. Your roof has:

1,658 hours of usable sunlight per year
Based on day-to-day analysis of weather patterns

708 sq feet available for solar panels
Based on 3D modeling of your roof and nearby trees

If your electric bill is at least $175/month, leasing solar panels could reduce it.

FINE-TUNE ESTIMATE  SEE SOLAR PROVIDERS

Wrong roof? Drag the marker to the right one.
MEDICAL IMAGING

Very good results using similar model for detecting diabetic retinopathy in retinal images
“Seeing” Go

Google’s AI just cracked the game that supposedly no computer could beat
By Mike Murphy | January 27, 2016

Google achieves AI 'breakthrough' at Go
An artificial intelligence program developed by Google beats Europe's top player at the ancient Chinese game of Go, about a decade earlier than expected.

Facebook trains AI to beat humans at Go

Google’s AI just cracked the game that supposedly no computer could beat.

Google's AI just cracked the game Go, which is considered one of the most challenging games for computers to handle due to the sheer number of possible moves a player can make at any given point. Until now, that is.
RankBrain in Google Search Ranking

Query: “car parts for sale”,
Doc: “Rebuilt transmissions …”

Deep Neural Network

Score for doc, query pair

Query & document features

Launched in 2015
Third most important search ranking signal (of 100s)

Bloomberg, Oct 2015: “Google Turning Its Lucrative Web Search Over to AI Machines”
Example: LSTM [Hochreiter et al, 1997][Gers et al, 1999]

\[
\begin{align*}
    i_t &= W_{ix} x_t + W_{ih} h_{t-1} + b_i \\
    j_t &= W_{jx} x_t + W_{jh} h_{t-1} + b_j \\
    f_t &= W_{fx} x_t + W_{fh} h_{t-1} + b_f \\
    o_t &= W_{ox} x_t + W_{oh} h_{t-1} + b_o \\
    c_t &= \sigma(f_t) \odot c_{t-1} + \sigma(i_t) \odot \tanh(j_t) \\
    h_t &= \sigma(o_t) \odot \tanh(c_t)
\end{align*}
\]

Enables long term dependencies to flow

```python
def __call__(self, inputs, state, scope=None):
    """Long short-term memory cell (LSTM)."""
    with vs.variable_scope(scope or type(self).__name__): # BasicLSTMCell
        # Parameters of gates are concatenated into one multiply for efficiency.
        c, h = array_ops.split(1, 2, state)
        concat = linear([inputs, h], 4 * self._num_units, True)

        # i = input_gate, j = new_input, f = forget_gate, o = output_gate
        i, j, f, o = array_ops.split(1, 4, concat)

        new_c = c * sigmoid(f + self._forget_bias) + sigmoid(i) * tanh(j)
        new_h = tanh(new_c) * sigmoid(o)

        return new_h, array_ops.concat(1, [new_c, new_h])
```

- TensorFlow
Sequence-to-Sequence Model

[Sutskever & Vinyals & Le NIPS 2014]

\[ P(y_1, \ldots, y_{T’} | x_1, \ldots, x_T) = \prod_{t=1}^{T’} p(y_t | v, y_1, \ldots, y_{t-1}) \]
Sequence-to-Sequence

● Active area of research
● Many groups actively pursuing RNN/LSTM
  ○ Montreal
  ○ Stanford
  ○ U of Toronto
  ○ Berkeley
  ○ Google
  ○ ...

● Further Improvements
  ○ Attention
  ○ NTM / Memory Nets
  ○ ...
Sequence-to-Sequence

- **Translation:** [Kalchbrenner et al., EMNLP 2013][Cho et al., EMLP 2014][Sutskever & Vinyals & Le, NIPS 2014][Luong et al., ACL 2015][Bahdanau et al., ICLR 2015]

- **Image captions:** [Mao et al., ICLR 2015][Vinyals et al., CVPR 2015][Donahue et al., CVPR 2015][Xu et al., ICML 2015]

- **Speech:** [Chorowsky et al., NIPS DL 2014][Chan et al., arxiv 2015]

- **Language Understanding:** [Vinyals & Kaiser et al., NIPS 2015][Kiros et al., NIPS 2015]

- **Dialogue:** [Shang et al., ACL 2015][Sordoni et al., NAAACL 2015][Vinyals & Le, ICML DL 2015]

- **Video Generation:** [Srivastava et al., ICML 2015]

How to do Image Captions?

$P(\text{English} \mid \text{French})$
**Image Captioning**

[Vinyals et al., CVPR 2015]

\[ \theta^* = \arg \max_{\theta} p(S|I) \]
Image Captions Research

*Human:* A young girl asleep on the sofa cuddling a stuffed bear.

*Model:* A close up of a child holding a stuffed animal.

*Model:* A baby is asleep next to a teddy bear.
Human: A man outside cooking with a sub in his hand.

BestModel: A man is holding a sandwich in his hand.

InitialModel: A man cutting a cake with a knife.
BestModel: A person is cooking some food on a grill.

Human: Someone is using a small grill to melt his sandwich.

InitialModel: A pizza sitting on top of a white plate.
Human: A woman holding up a yellow banana to her face.

BestModel: A woman holding a banana up to her face.

InitialModel: A close up of a person eating a hot dog.
Human: A blue, yellow and red train travels across the tracks near a depot.

BestModel: A blue and yellow train traveling down train tracks.

InitialModel: A train that is sitting on the tracks.
A man holding a tennis racquet on a tennis court.

Two pizzas sitting on top of a stove top oven.

A group of young people playing a game of Frisbee.

A man flying through the air while riding a snowboard.
Pointer Networks

➢ Goal: Mappings where outputs are (sub)sets of inputs
➢ Travelling Salesman Problem

➢ Convex Hulls

[Vinyals, Fortunato & Jaitly, NIPS 2015]
Pointer Networks

[Vinyals, Fortunato & Jaitly, NIPS 2015]
Neural Conversational Models

- Take movie subtitles (~900M words) or IT HelpDesk chats
- Predict the next dialog from history

i got to go.
no.
i get too emotional when i drink.
have another beer. i 've got to get up early.
no, you don 't. sit down.
i get too emotional when i drink.
will you have another beer?
i 've got to go!
why?
i got to get up early in the morning.
you 're drunk.
and emotional!
you got to go.

[Vinyals & Le ICML DL Workshop 2015]
Smart Reply

April 1, 2009: April Fool’s Day joke

Nov 5, 2015: Launched Real Product

Feb 1, 2016: >10% of mobile Inbox replies
Smart Reply

Incoming Email

dcorrado
to me

5:37 PM:

Hi all,
We wanted to invite you to join us for an early
Thanksgiving on November 22nd, beginning
around 2PM. Please bring your favorite dish! RSVP by
next week.

Dave

Small Feed-Forward Neural Network

Activate Smart Reply?

yes/no
Smart Reply

Incoming Email

Hi all,
We wanted to invite you to join us for an early Thanksgiving on November 22nd, beginning around 2PM. Please bring your favorite dish! RSVP by next week.
Dave

Small Feed-Forward Neural Network

Deep Recurrent Neural Network

Generated Replies

Activate Smart Reply?

yes/no

Google Research Blog
- Nov 2015
Example: LSTM

for i in range(20):
    m, c = LSTMCell(x[i], mprev, cprev)
    mprev = m
    cprev = c
Example: Deep LSTM

for i in range(20):
    for d in range(4):  # d is depth
        input = x[i] if d is 0 else m[d-1]
        m[d], c[d] = LSTMCell(input, mprev[d], cprev[d])
        mprev[d] = m[d]
        cprev[d] = c[d]
Example: Deep LSTM

for i in range(20):
    for d in range(4): # d is depth
        input = x[i] if d is 0 else m[d-1]
        m[d], c[d] = LSTMCell(input, mprev[d], cprev[d])
        mprev[d] = m[d]
        cprev[d] = c[d]
Example: Deep LSTM

for i in range(20):
    for d in range(4): # d is depth
        with tf.device("/gpu:%d" % d):
            input = x[i] if d is 0 else m[d-1]
            m[d], c[d] = LSTMCell(input, mprev[d], cprev[d])
            mprev[d] = m[d]
            cprev[d] = c[d]
This is very big!

Split softmax into 4 GPUs

1000 LSTM cells
2000 dims per timestep

2000 x 4 = 8k dims per sentence

80k softmax by 1000 dims
This is very big!

Split softmax into 4 GPUs

1000 LSTM cells
2000 dims per timestep

2000 x 4 = 8k dims per sentence
80k softmax by 1000 dims
This is very big!

Split softmax into 4 GPUs

1000 LSTM cells
2000 dims per timestep

2000 x 4 = 8k dims per sentence
80k softmax by 1000 dims
This is very big!

Split softmax into 4 GPUs

1000 LSTM cells
2000 dims per timestep

2000 x 4 = 8k dims per sentence
80k softmax by 1000 dims
This is very big!

Split softmax into 4 GPUs

1000 LSTM cells
2000 dims per timestep

2000 x 4 = 8k dims per sentence
This is very big!

Split softmax into 4 GPUs

1000 LSTM cells, 2000 dims per timestep

2000 x 4 = 8k dims per sentence
80k softmax by 1000 dims
This is very big!

Split softmax into 4 GPUs

1000 LSTM cells
2000 dims per timestep

2000 x 4 = 8k dims per sentence
80k softmax by 1000 dims
This is very big!
Split softmax into 4 GPUs
1000 LSTM cells
2000 dims per timestep
2000 x 4 = 8k dims per sentence
This is very big!

Split softmax into 4 GPUs

1000 LSTM cells
2000 dims per timestep

2000 x 4 = 8k dims per sentence
This is very big!

Split softmax into 4 GPUs

1000 LSTM cells
2000 dims per timestep
2000 x 4 = 8k dims per sentence
80k softmax by 1000 dims
This is very big!
Split softmax into 4 GPUs
1000 LSTM cells
2000 dims per timestep
2000 x 4 = 8k dims per sentence
80k softmax by 1000 dims
This is very big!

Split softmax into 4 GPUs

1000 LSTM cells
2000 dims per timestep

2000 x 4 = 8k dims per sentence
TensorFlow Queues

Input prefetching

Grouping similar examples

Randomization/Shuffling

Enqueue

Dequeue

Queue
Example: Deep LSTMs

- Wrinkles
  - Bucket sentences by length using a queue per length
  - Dequeue when a full batch of same length has accumulated
  - N different graphs for different lengths
  - Alternative: while loop
We use the ReplicaDeviceSetter() device function to automatically assign Variables to the 'ps' jobs.

```python
with tf.device("/cpu:0"):
    # Create the Mnist model.
    model = MnistModel(batch_size=16, hidden_units=200)

    # Get an initialized, and possibly recovered session.
    sess = tf.Session()

    # Train the model.
    for local_step in xrange(FLAGS.max_steps):
        _, loss, step = sess.run([model.train_op, model.loss, model.global_step])
        if local_step % 1000 == 0:
            print "step %d: %g" % (step, loss)
```
We use the ReplicaDeviceSetter() device function to automatically assign Variables to the 'ps' jobs.

```python
with tf.device(tf.ReplicaDeviceSetter(parameter_devices=10)):
    # Create the Mnist model.
    model = MnistModel(batch_size=16, hidden_units=200)

# Create a Supervisor. It will take care of initialization, summaries,
# checkpoints, and recovery. When multiple replicas of this program are running,
# the first one, identified by --task=0 is the 'chief' supervisor (e.g., initialization, saving)
supervisor = tf.Supervisor(is_chief=(FLAGS.task == 0), saver=model.saver)

# Get an initialized, and possibly recovered session.
sess = supervisor.PrepareSession(FLAGS.master_job)

# Train the model.
for local_step in xrange(int32_max):
    _, loss, step = sess.run([model.train_op, model.loss, model.global_step])
    if step >= FLAGS.max_steps:
        break
    if local_step % 1000 == 0:
        print "step %d: %g" % (step, loss)"
Combining Vision with Robotics


“Learning Hand-Eye Coordination for Robotic Grasping with Deep Learning and Large-Scale Data Collection”, Sergey Levine, Peter Pastor, Alex Krizhevsky, & Deirdre Quillen, Arxiv, arxiv.org/abs/1603.02199
Network Optimizations

- Neural net training very tolerant of reduced precision
- e.g. drop precision to 16 bits across network
Network Optimizations

- Neural net training very tolerant of reduced precision
- e.g. drop precision to 16 bits across network
Quantization for Inference

- Need even less precision for inference
- 8-bit fixed point works well, but many ways of quantizing
- Critical for things like mobile devices
  - w/quantization, high-end smart phone can run Inception model at >6 frames per second (fps)
How Can You Get Started with Machine Learning?

Three ways, with varying complexity:

(1) Use a Cloud-based API (Vision, Speech, etc.)
(2) Use an existing model architecture, and retrain it or fine tune on your dataset
(3) Develop your own machine learning models for new problems

More flexible, but more effort required
Use Cloud-based APIs

GOOGLE TRANSLATE API
Dynamically translate between thousands of available language pairs
cloud.google.com/translate

CLOUD SPEECH API
Speech to text conversion powered by machine learning
cloud.google.com/speech

CLOUD VISION API
Derive insight from images with our powerful Cloud Vision API
cloud.google.com/vision

CLOUD TEXT API
Use Cloud Text API for sentiment analysis and entity recognition in a piece of text.
cloud.google.com/text
Use Cloud-based APIs

- **GOOGLE TRANSLATE API**
  - Dynamically translate between thousands of available language pairs
  - [cloud.google.com/translate](http://cloud.google.com/translate)

- **CLOUD SPEECH API**
  - Speech to text conversion powered by machine learning
  - [cloud.google.com/speech](http://cloud.google.com/speech)

- **CLOUD VISION API**
  - Derive insight from images with our powerful Cloud Vision API
  - [cloud.google.com/vision](http://cloud.google.com/vision)

- **CLOUD TEXT API**
  - Use Cloud Text API for sentiment analysis and entity recognition in a piece of text.
  - [cloud.google.com/text](http://cloud.google.com/text)
Google Cloud Vision API

https://cloud.google.com/vision/

"running", "score": 0.99803412,
"marathon", "score": 0.99482006

"joyLikelihood": "VERY_LIKELY"

,"description": "ABIERTO
", "local": "es"
Managed scalable machine learning platform

Google Cloud Machine Learning is a managed platform that enables you to easily build machine learning models that work on any type of data, of any size. Create your model with the powerful TensorFlow framework, that powers many Google products from Google Photos, to Google Cloud Speech. Build models of any size with our...
A Few TensorFlow Community Examples
(From more than 2000 results for ‘tensorflow’ on GitHub)

- DQN: github.com/nivwusquorum/tensorflow-deepq
- NeuralArt: github.com/woodrush/neural-art-tf
- Char RNN: github.com/sherjilozair/char-rnn-tensorflow
- Keras ported to TensorFlow: github.com/fchollet/keras
- Show and Tell: github.com/jazzsaxmafia/show_and_tell.tensorflow
- Mandarin translation: github.com/jikexueyuanwiki/tensorflow-zh

...
A Few TensorFlow Community Examples
(From more than 2100 results for ‘tensorflow’ on GitHub)

- DQN: github.com/nivwusquorum/tensorflow-deepq
- NeuralArt: github.com/woodrush/neural-art-tf
- Char RNN: github.com/sherjilozair/char-rnn-tensorflow
- Keras ported to TensorFlow: github.com/fchollet/keras
- Show and Tell: github.com/jazzsaxmafia/show_and_tell.tensorflow
- Mandarin translation: github.com/jikexueyuanwiki/tensorflow-zh

...
Reinforcement Learning using Tensor Flow

Quick start

Check out Karpathy game in notebooks folder.

The image above depicts a strategy learned by the DeepQ controller. Available actions are accelerating top, bottom, left or right. The reward signal is +1 for the green fellas, -1 for red and -5 for orange.
"Neural Art" in TensorFlow

An implementation of "A neural algorithm of Artistic style" in TensorFlow, for

- Introductory, hackable demos for TensorFlow, and
- Demonstrating the use of importing various Caffe cnn models (VGG and illustration2vec) in TF.

In this work, I put effort in putting the code simple as possible, for being a good introductory code to TF. For this reason, I also implemented very basic uses of TensorBoard (the visualizer). I also aimed on demonstrating the use of importing various Caffe models from *.caffemodel files into TensorFlow, especially models that seemed not to be imported by anybody yet in TF (as far as I know). Based on https://github.com/ethereon/caffe-tensorflow, I modified the importer so that it can import illustration2vec (http://illustration2vec.net/), which is another CNN available as a Caffe model. Using different CNNs yields different results, which reflects the characteristics of the model.

In the Neural Art problem setting, the weights of the CNN are fixed, and the input image into the CNN is the only "trainable" variable, making the code easy to understand (the optimized/trained image is the output image). I hope this example serves as a good introduction to TensorFlow as well as for entertainment purposes.
GitHub repository: [github.com/sherjilozair/char-rnn-tensorflow](https://github.com/sherjilozair/char-rnn-tensorflow)

**char-rnn-tensorflow**

Multi-layer Recurrent Neural Networks (LSTM, RNN) for character-level language models in Python using Tensorflow.

Inspired from Andrej Karpathy's [char-rnn](https://github.com/andrejkarpathy/char-rnn).

**Requirements**

- Tensorflow

**Basic Usage**

To train with default parameters on the tinyshakespeare corpus, run `python train.py`.

To sample from a checkpointed model, `python sample.py`.
Keras: Deep Learning library for Theano and TensorFlow

You have just found Keras.

Keras is a minimalist, highly modular neural networks library, written in Python and capable of running either on top of either TensorFlow or Theano. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result with the least possible delay is key to doing good research.

Use Keras if you need a deep learning library that:

- allows for easy and fast prototyping (through total modularity, minimalism, and extensibility).
- supports both convolutional networks and recurrent networks, as well as combinations of the two.
- supports arbitrary connectivity schemes (including multi-input and multi-output training).
- runs seamlessly on CPU and GPU.

Read the documentation at Keras.io.

Keras is compatible with: - Python 2.7-3.5 with the Theano backend - Python 2.7 with the TensorFlow backend
Neural Caption Generator

  - Borrowed some code and ideas from Andrej Karpathy’s NeuralTalk.
- You need flickr30k data (images and annotations)

Code

- `make_flickr_dataset.py`: Extracting feats of flickr30k images, and save them in `./data/feats.npy`
- `model_tensorflow.py`: TensorFlow Version
- `model_theano.py`: Theano Version

Usage

- Flickr30k Dataset Download
- Extract VGG Features of Flicker30k images (`make_flickr_dataset.py`)
- Train: run `train()` in `model_tensorflow.py` or `model_theano.py`
- Test: run `test()` in `model_tensorflow.py` or `model_theano.py`
  - parameters: VGG FC7 feature of test image, trained model path
TensorFlow is an Open Source Software Library for Machine Intelligence

你正在翻译的项目可能会比 Android 系统更加深远地影响着世界！

缘起

2015年11月9日，Google 官方在其博客上称，Google Research 宣布推出第二代机器学习系统 TensorFlow，针对先前的 DistBelief 的短板有了各方面的加强，更重要的是，它是开源的，任何人都可以用。

机器学习作为人工智能的一种类型，可以让软件根据大量的数据来对未来的情况进行阐述或预判。如今，领先的科技巨头无不在这机器学习下予以极大投入。Facebook、苹果、微软，甚至国内的百度。Google 自然也在其中。 「TensorFlow」是 Google
Concluding Remarks

● Model and Data Parallelism enable great ML work:
  ○ Neural Machine Translation: ~6x speedup on 8 GPUs
  ○ Inception / Imagenet: ~40x speedup on 50 GPUs
  ○ RankBrain: ~300X speedup on 500 machines
● TensorFlow open-source community vibrant and growing
● TensorFlow makes it easy to express ML computations
What Does the Future Hold?

Deep learning usage will continue to grow and accelerate:

- Across more and more fields and problems:
  - robotics, self-driving vehicles, ...
  - health care
  - video understanding
  - dialogue systems
  - personal assistance
  - ...

Google Brain Residency Program

One year immersion program in deep learning research
- First class started six weeks ago, planning for next year’s class is underway

Learn to conduct deep learning research w/experts in our team
- Fixed one-year employment with salary, benefits, ...
- Goal after one year is to have conducted several research projects
- Interesting problems, TensorFlow, and access to computational resources

g.co/brainresidency
Google Brain Residency Program

Who should apply?

- people with BSc, MSc or PhD, ideally in CS, mathematics or statistics
- completed coursework in calculus, linear algebra, and probability, or equiv.
- programming experience
- motivated, hard working, and have a strong interest in deep learning

[g.co/brainresidency]
Google Brain Residency Program

Current class for June 2016 to May 2017

- $\frac{1}{3}$ B.S, $\frac{1}{3}$ M.S., $\frac{1}{3}$ Ph.D. or postdoc
- $\frac{1}{2}$ coming straight from school, $\frac{1}{2}$ with some post-school working experience
- Mix of backgrounds: computer scientists, math/stats, EE, physics, comp bio, ...

Applications for class for June 2017 to May 2018 will open in Fall 2016

g.co/brainresidency
Further Reading

- Sutskever, Vinyals, & Le, Sequence to Sequence Learning with Neural Networks, NIPS, 2014, arxiv.org/abs/1409.3215.
- TensorFlow white paper, tensorflow.org/whitepaper2015.pdf (clickable links in bibliography)

g.co/brain (We’re hiring! Also check out Brain Residency program at g.co/brainresidency)
www.tensorflow.org
research.google.com/people/jeff
research.google.com/pubs/BrainTeam.html

Questions?